第5章

ソフト・マクロのCPUの実装法

――PicoBlazeを使った設計を体験する

田中良平

ここでは，無償で利用可能な８ビットCPUコアである米国Xilinx社の「PicoBlaze」を，本誌付属のFPGA基板で利用する方法について解説する．ISE WebPACKを使ってPicoBlazeを組み込む方法のほか，PicoBlaze用のソフトウェア開発などについても取り上げる．なお，PicoBlazeのアーキテクチャの概要については，前号の記事（2004年12月号，pp.32-37）を参照していただきたい．（編集部）

ここでは，米国Xilinx社のPicoBlaze向けサンプル回路を付属FPGA基板へ実装する過程について解説します．本稿によって，

- ISE WebPACKを使用したPicoBlazeの組み込みかた
- PicoBlaze制御用のソフトウェアのプログラミング
- PicoBlaze開発ツールの使いかた
- PicoBlazeによるシリアル通信機能の実現方法
- GUI開発ツールの使いかた

をご理解いただけると思います．

PicoBlazeを付属FPGA基板に搭載されているXC3S50に実装しても，まだ80％の論理ブロックが余ります．要求に応じてさまざまな回路を実装してみるとよいでしょう．

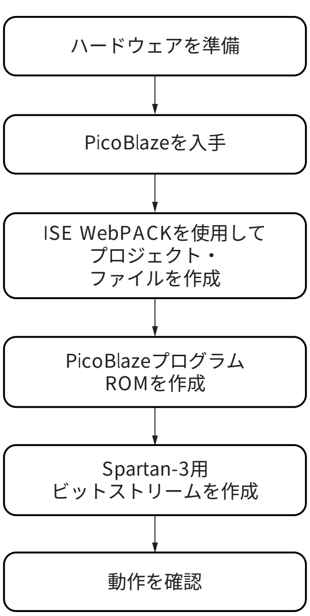
1. 開発環境の準備

PicoBlazeを使うための流れを，図1に示します．今回は，本誌付属のFPGA基板でPicoBlazeを動作させます．表1に示すものを用意しておく必要があります．

● ハードウェアを準備

PicoBlazeは，実装するFPGAファミリごとにコアや開発ツールが異なります．そこでまず，動作させるハードウェアを決定しておく必要があります．

本稿をお読みいただいている方であれば，お手元にSpartan-3が搭載された本誌付属のFPGA基板があると思います．ですから，動作させるFPGAはSpartan-3ということになります．



◀図1 開発の流れ

▶表1 使用する開発環境

アイテム	入手方法など
FPGAボード	本誌付属のFPGA基板（Spartan-3搭載）を利用，シリアル・ポートやクロック，LED，電源を追加（今回は，別のFPGA評価ボードに搭載されているものを活用）
コンフィグレーション・ツール	Xilinx社のParallel Cable IVを利用，Xilinx社の代理店で購入可能
PicoBlaze	Xilinx社のホームページから無償でダウンロード可能
ISE WebPACK	本誌付属CD-ROMに収録，Xilinx社のホームページから無償でダウンロード可能
pBlazIDE	Mediatronix社のホームページから無償でダウンロード可能

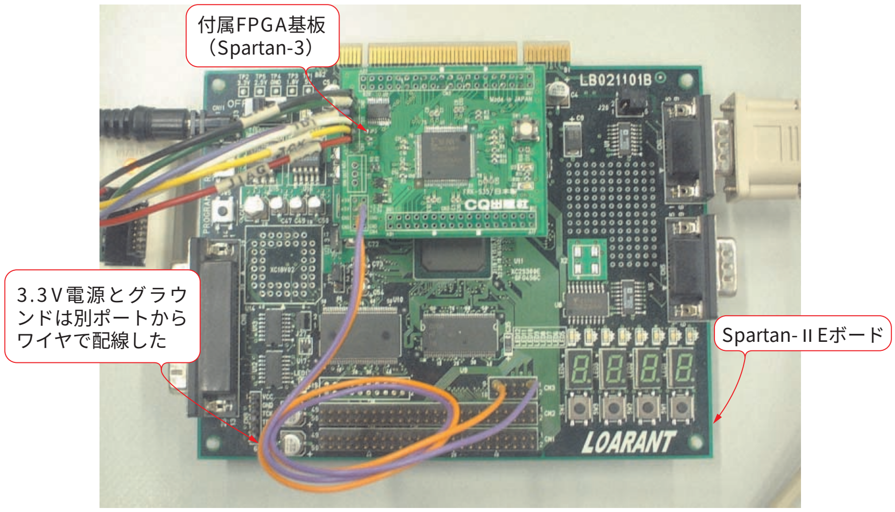
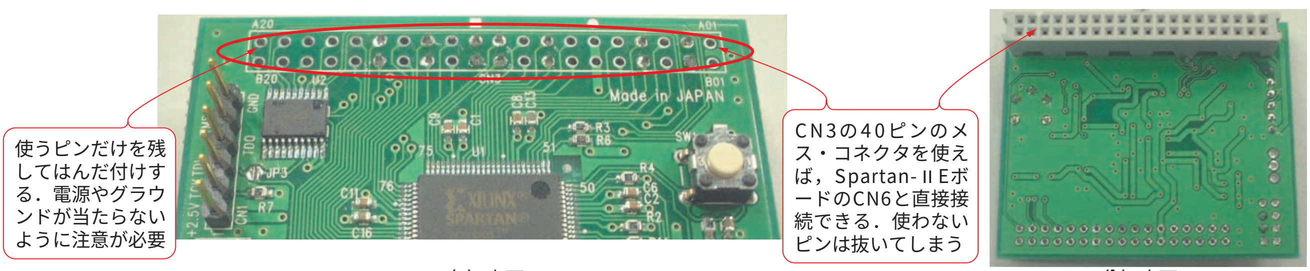


写真1
動作させるFPGAボード
付属FPGA基板以外に必要な回路については，「Spartan-ⅡE300評価キット」のSpartan-ⅡEボードに搭載されている回路を利用した．Spartan-ⅡEはただの配線になっている．



（a）表面　（b）裏面

写真2　付属FPGA基板に取り付けたコネクタ
実際に使用するピンだけにして，はんだ付けしている．

付属FPGA基板でPicoBlazeを使うためには，シリアル・ポートやクロックなどが必要になります．また，周辺回路を製作する必要があります．はんだ付けに慣れていない方にとっては，この製作がたいへんに思えてしまうかもしれません．そこで周辺回路の製作を少しでも楽にする方法を紹介します．これは，ほかのFPGAボードなどに搭載された電源や入出力回路を流用するという方法です．これなら基板間を接続するためのコネクタをはんだ付けする程度ですみます．筆者も今回はこの方法を採りました（**写真1**）．

40ピンのメス・コネクタを用意し，使用しないピンを抜き取ってしまいます．これは，付属FPGA基板と周辺回路として使うSpartan-ⅡEボードのコネクタのピン配置が異なるからです．実際に使用するピンだけにして，はんだ付けします（**写真2**）．電源はワイヤで別供給としました．Spartan-ⅡEボード側のFPGAには，Spartan-ⅡEボード上のシリアル・ポートとクロック（66MHz），LEDをSpartan-3に接続するための回路（事実上ただの配線）を組み込みます（付属CD-ROMに収録しているroute.bitを使用）．

回路を動作させるハードウェア以外にも必要なものがあります．Spartan-3に回路データを書き込むためのコンフィグレーション・ツール（Parallel Cable IVなど）です．

● PicoBlazeを入手

PicoBlazeは，Xilinx社のホームページ（http://www.xilinx.co.jp/）から無償でダウンロードできます．ホームから「製品とサービス」を選択すると，「ザイリンクス製品とサービス」のページになります．「デザイン リソース」の欄にある「プロセッサ セントラル」を選びます．「プロセッサ セントラル」のページで，「ソフト プロセッサ」の欄にある「PicoBlaze」を選択すると，PicoBlazeのページが開きます．

PicoBlazeのページからは，**図2**の手順に従って進んでください．途中で入力が求められるユーザIDとパスワードは，これまでXilinx社のFPGAを使っていた方ならすでにお持ちだと思います．今回初めてXilinx社のFPGAを使わ

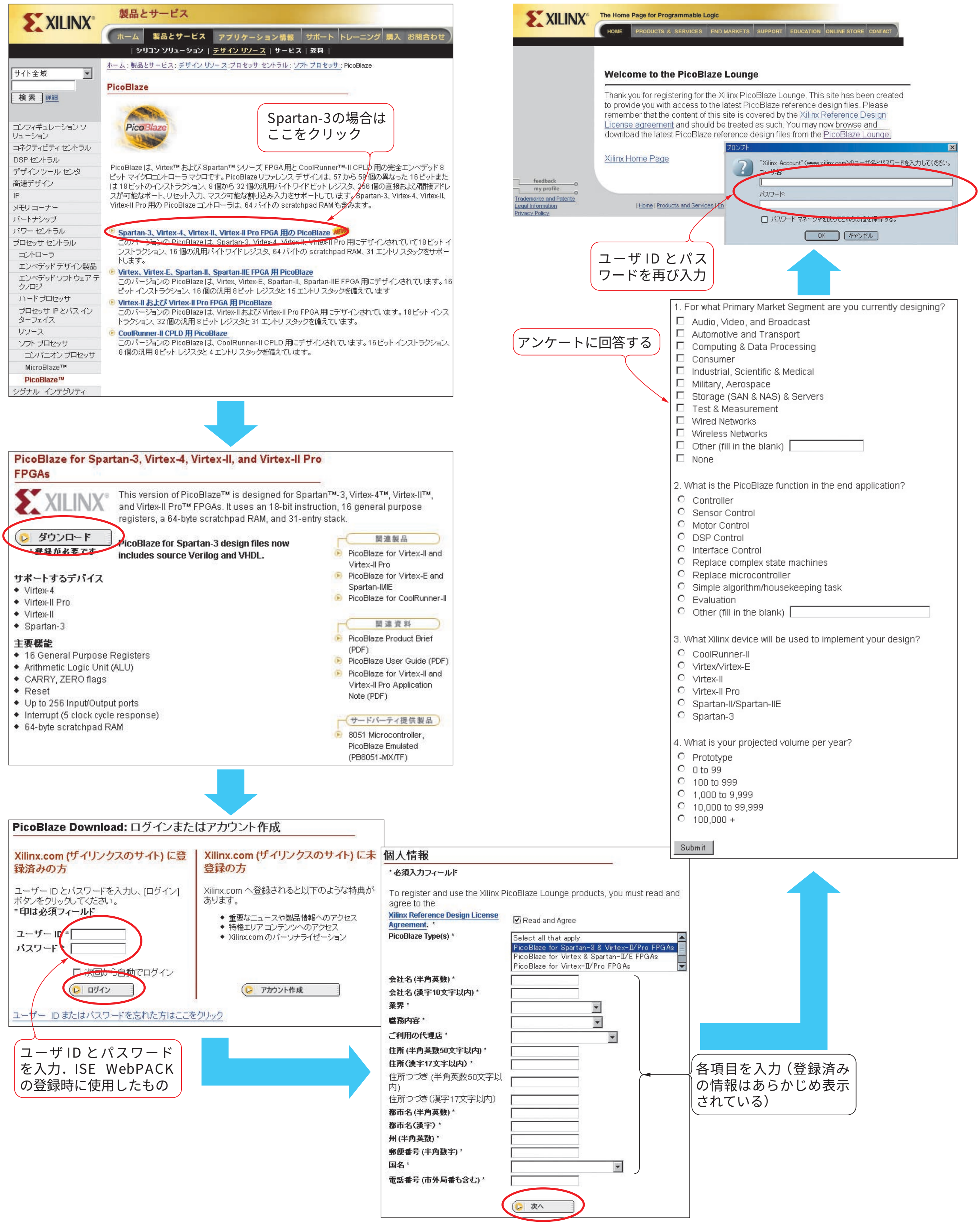


図2　PicoBlazeの入手

PicoBlazeのページ（http://www.xilinx.com/products/design_resources/proc_central/grouping/picoblaze.htm）からダウンロードできる．

れる方も，ISE WebPACKのインストールにあたって登録しているはずです（本特集の第4章を参照）．ログイン後，簡単なアンケートに答えると，図3のPicoBlazeラウンジへ進むことができます．

PicoBlazeファイルは，PicoBlazeラウンジからダウンロードできます．ファイルはZIP形式です．WinZIPなどのツールで解凍すると，VHDLおよびVerilog HDLのソース・コード，ソフトウェア開発用のアセンブラ，JTAGローダ注，各種ドキュメント類が得られます．

PicoBlazeラウンジには，ほかにも便利な開発ツールが登録されているPicoBlaze User Resourcesがあります．またユーザどうしの情報交換の場として活用されているPicoBlaze Forumへもここを起点としてアクセスできるので，ブックマーク登録を行っておくと便利です．



図3　PicoBlazeラウンジ
このページから設計ファイルやサンプル・デザイン，サード・パーティの開発ツールなどをダウンロードできるPicoBlaze User Resourcesへアクセスできる．

● pBlazIDEの入手

pBlazIDEは，オランダMediatronix社のGUIによるソフトウェア開発ツールです．Mediatronix社のホームページから無償でダウンロードできます．

図4のように，ホームから「Tools」を選択します．ツールの一覧が表示されるので，「pBlazIDE」を選びます．pBlazIDEのページの下のほうに，ダウンロードのリンクがあります．

ファイルはZIP形式です．WinZIPなどのツールで解凍すると，EXE形式の実行ファイルになります．

注：実際のFPGA上でPicoBlazeを検証する場合にプログラムの変更が発生すると，アセンブル（ROMファイル作成）→論理合成および配置配線→ビットストリームのダウンロードという手順を踏むため，手間と時間がかかる．プログラム・リセットのために，JTAGローダではプログラムROMに若干の修正と，PicoBlazeとの接続を変更する必要がある．しかし，いったんビットストリームを作成しておけば，プログラム変更の際にJTAG経由でBlock RAM内のプログラム・データのみアップデートすることが可能となる．つまり，アセンブル（ROMファイル作成）→プログラム・ダウンロードの手順になるため，論理合成や配置配線の手間から解放される．詳しくはJTAG loaderフォルダ内のJTAG_loader_quick_guide.pdfを参照してほしい．

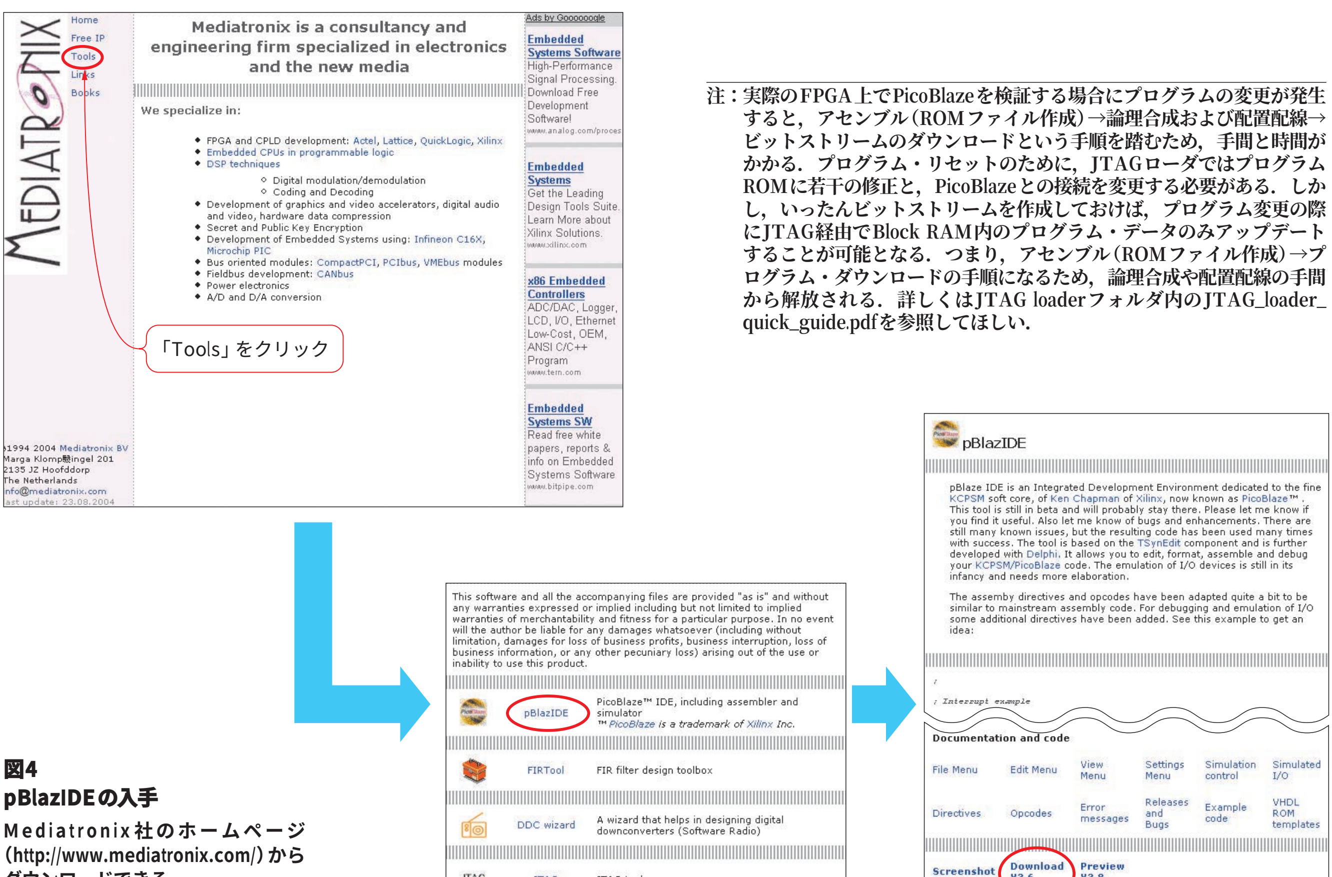


図4
pBlazIDEの入手
Mediatronix社のホームページ（http://www.mediatronix.com/）からダウンロードできる．

2. サンプル回路を動かしてみる

PicoBlazeのファイルに標準で用意されているサンプル・デザインであるUART(universal asynchronous receiver-transceiver)時計を付属のFPGA基板へ組み込むまでを解説します.

● UART時計の概要

UART時計は，パソコンなどとのシリアル通信によって時刻設定や時刻表示，アラーム設定を行える時計です．PicoBlazeコア，送信用UART，受信用UART，タイマ，プログラム・メモリ(内蔵メモリ・ブロックのBlock RAMを使用)から構成されます(図5).

PicoBlazeは8ビットのCPUコアなので，周辺の機能ブロックをPicoBlazeからソフトウェア制御することができます．プログラミング言語はアセンブラのみとなります．1命令を2クロックで実行します．今回はソフトウェアの詳細については解説しませんが，実機でいろいろとサンプル・プログラムを動かしたり，開発ツール上でコード・シミュレーションを実施すればだいたい理解できると思います．また，CPUコアについての理解が深まれば，さまざまな新しい発見があると思います.

● ISE WebPACKによる設計フロー

FPGAの設計では，Xilinx社のFPGA設計ツール「ISE WebPACK」を使用します．PicoBlazeを使うときも同じです.

1) 設計ファイルの登録

ISE WebPACK(Project Navigator)を起動した後，新規プロジェクトの作成を行ってHDLソース・ファイルを登録します．今回は，図6のように，サンプルとして用意されているUART時計の設計データを使います.

プロジェクト作成の手順を図7に示します．PicoBlazeのコアは，VHDLとVerilog HDLのソース・コードで提供されているので，通常のHDL設計のときの操作となんら変わりません.

2) ソフトウェアの開発

PicoBlazeはCPUであるため，動作させるにはソフトウェアを作成する必要があります．今回は，サンプルとして提供されているソース・コードを使います．サンプル・コードの入っているフォルダ(ここではVHDLフォルダとする)とアセンブラ実行ファイル(KCPSM3.exe)が含まれているAssemblerフォルダをプロジェクト・フォルダへコピー(ここではD:\PicoBlaze\UART_CLOCK)します.

サンプル・ソフトウェアが想定しているハードウェアと，実際に動作させるハードウェアの仕様が異なるため，若干の修正が必要です．今回実装するハードウェア仕様に合わ

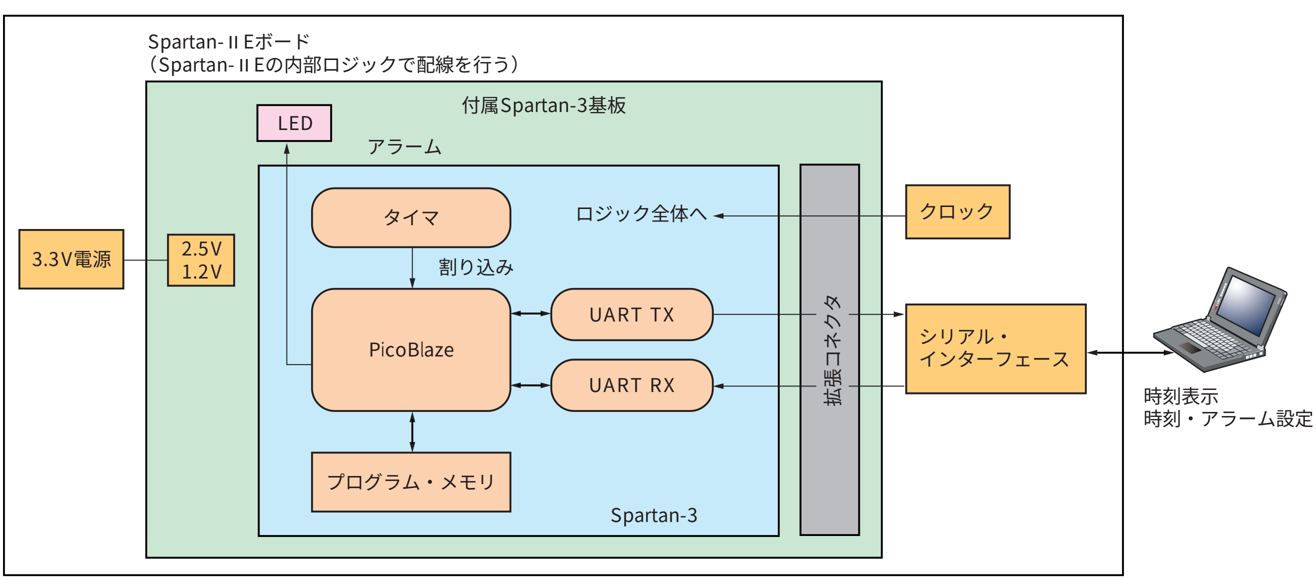


図5　UART時計のブロック図

シリアル・インターフェース，クロック，3.3V電源はSpartan-ⅡEボードのものを使う．Spartan-3の回路は，PicoBlazeコア，送信用UART，受信用UART，タイマ，プログラム・メモリ(内蔵メモリ・ブロックのBlock RAMを使用)で構成されている.

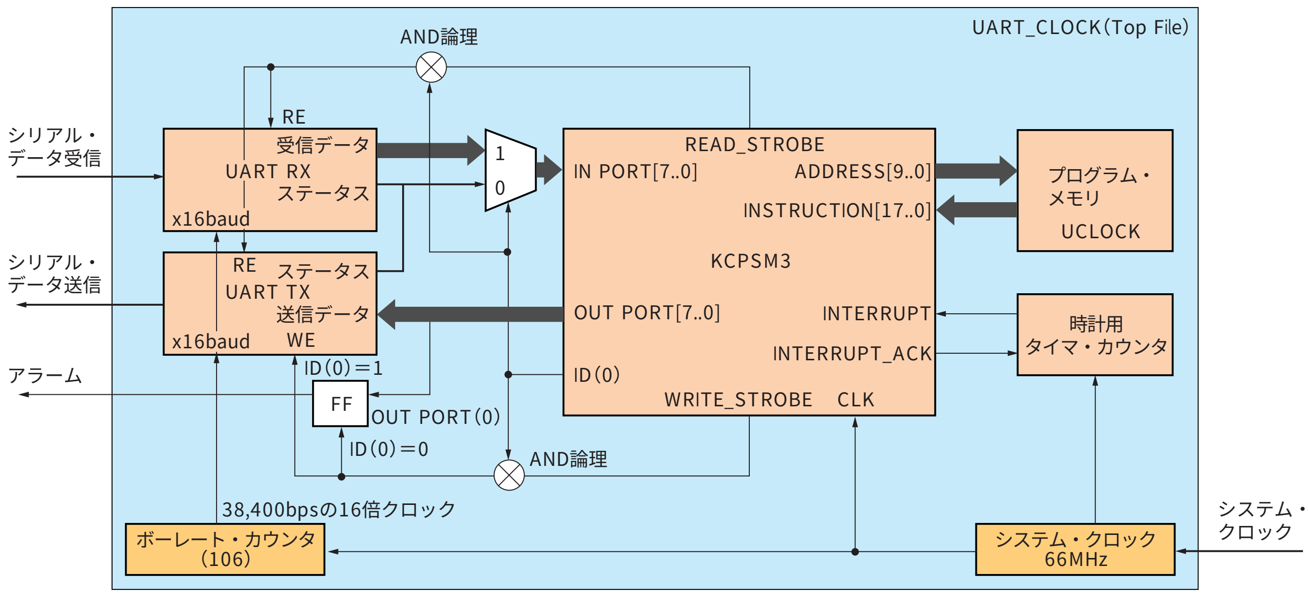


図6　UART時計のトップ・ブロック

UARTモジュールは送信用と受信用に分離されている．UARTモジュール，アラーム・レジスタはそれぞれPicoBlazeのIN，OUTポートに接続され，ID番号によってアクセスする周辺モジュールの選択が可能な構成となっている．タイマ・モジュールが割り込みを使用しており，このサンプル・デザインによってひととおりの使いかたを知ることができる．

5

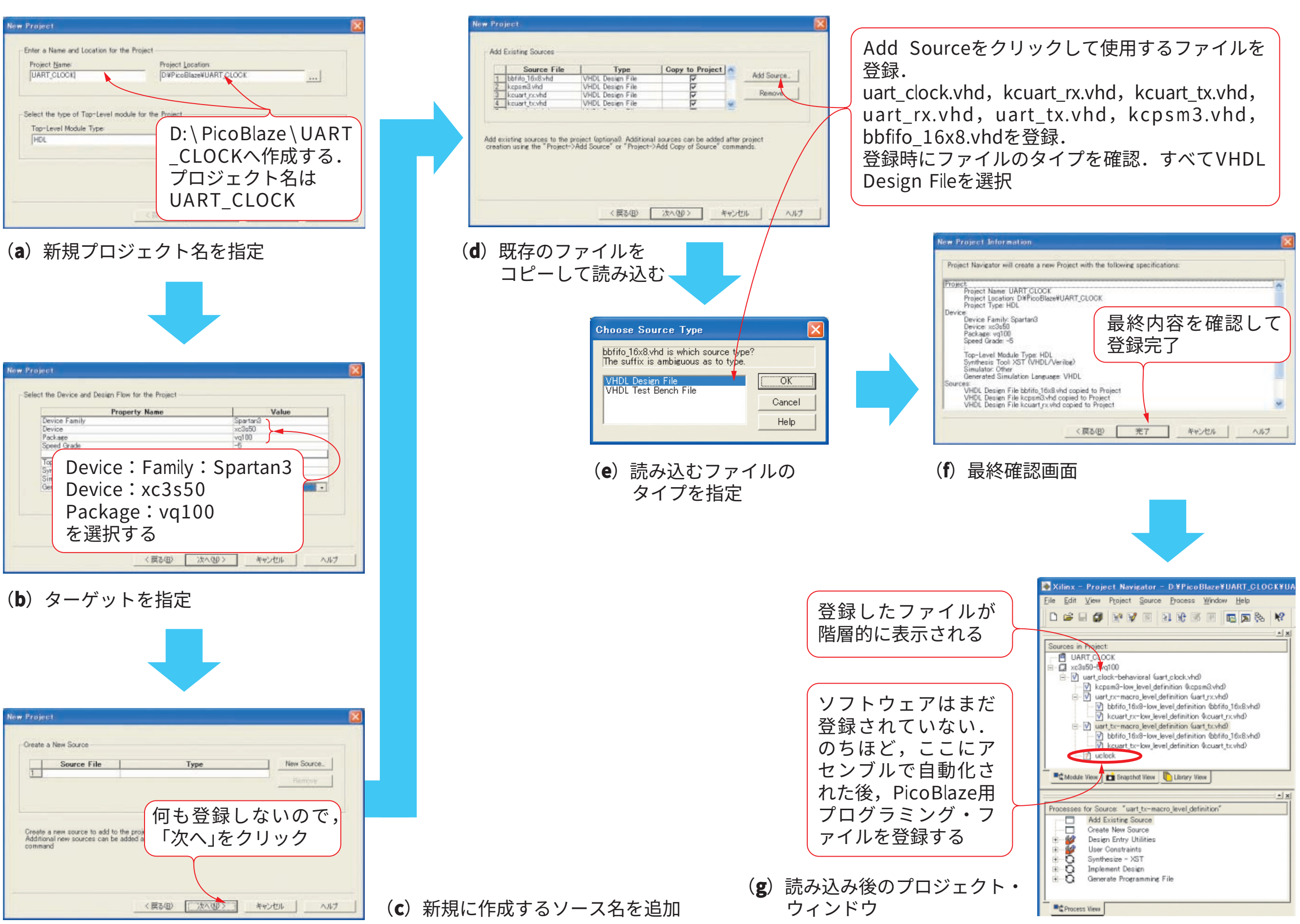


図7　プロジェクトの作成

リスト1　シリアル通信速度の修正

uart_clock.vhdの333行目．

```
--
-- Set baud rate to 38400 for the UART communications
-- Requires en_16_x_baud to be 614400Hz which is a single cycle pulse every 107 cycles at 66MHz
--
-- NOTE : If the highest value for baud_count exceeds 127 you will need to adjust
--        the range of integers in the signal declaration for baud_count.
--

baud_timer: process(clk)
begin
  if clk'event and clk='1' then
    if baud_count=106 then
        baud_count <= 0;
      en_16_x_baud
                 <= '1';
     else
        baud_count
      <= baud_count + 1;
      en_16_x_baud <= '0';
    end if;
  end if;
end process baud_timer;
```

使用するシステム・クロックに合わせて，en_16_xbaudが通信速度の16倍パルスになるようにカウンタ値を設定

```
-- Interrupt is a generated once every 66 clock cycles to provide a 1us reference.
-- Interrupt is automatically cleared by interrupt acknowledgment from KCPSM3.
--

Timer: process(clk)
begin

  if clk'event and clk='1' then

    if timer_count=65 then
       timer_count <= 0;
       timer_pulse <= '1';
     else
       timer_count
         <= timer_count + 1;
       timer_pulse <= '0';
    end if;

    if interrupt_ack = '1'
                     then
       interrupt <= '0';
     elsif timer_pulse = '1'
                     then
       interrupt <= '1';
      else
       interrupt <= interrupt;
    end if;

  end if;

end process Timer;
```

システム・クロックとして66MHzを使用しているので，1μsのタイミングを作るにはカウンタ値を66とする（0～65をカウント）

▶ **リスト2**
タイマのカウント時間の設定

uart_clock.vhdの194行目．

```
  -- adding the output registers to the clock
processor

  output_ports: process(clk)
  begin

    if clk'event and clk='1' then
      if write_strobe='1' then

        -- Alarm register at address 00 hex with
data bit0 providing control

        if port_id(0)='0' then
          alarm <= not out_port(0);
        end if;

      end if;

    end if;
```

LED論理のつごう上，反転するように変更

▶ **リスト3**
LED表示のための修正

uart_clock.vhdの276行目．

せて，**リスト1**～**リスト3**に示す修正を施しました．

リスト1は，シリアル通信速度の修正です．今回使用する66MHzのシステム・クロックで38,400bps通信を行うため，ボーレート・カウンタの値を変更します．使用するクロックに応じて以下の式で計算し直します．

カウンタ値＝システム周波数÷通信速度÷16

リスト2は，タイマのカウント時間の設定です．66MHzのシステム・クロックが66MHzを使用しているので，時間を測定する際の基準となるカウンタ値も変更する必要があります．

リスト3は，アラームをLED表示で代用するための修正です．付属FPGA基板上に実装されたLEDは，FPGAのI/Oピンが“L”のときに点灯します．

3）ソフトウェア・シミュレーション

ソフトウェアを実機動作で確認する前に，pBlazIDEのコード・シミュレーション機能を使ってプログラムのふるまいを見ます．

詳細はpBlazIDEを実際に使いながら確認していただくとして，ここではKCPSM3用に記述されているアセンブリ・コード（uclock.psm）をインポートして使用します．

まず，pBlazIDEを起動し，「File」→「Import」を選択し，uclock.psmをインポートします．すると，**図8**のように記述方式もpBlazIDE用に自動変換して取り込まれていることがわかります．

ファイル出力とシミュレーションを実行するためには，

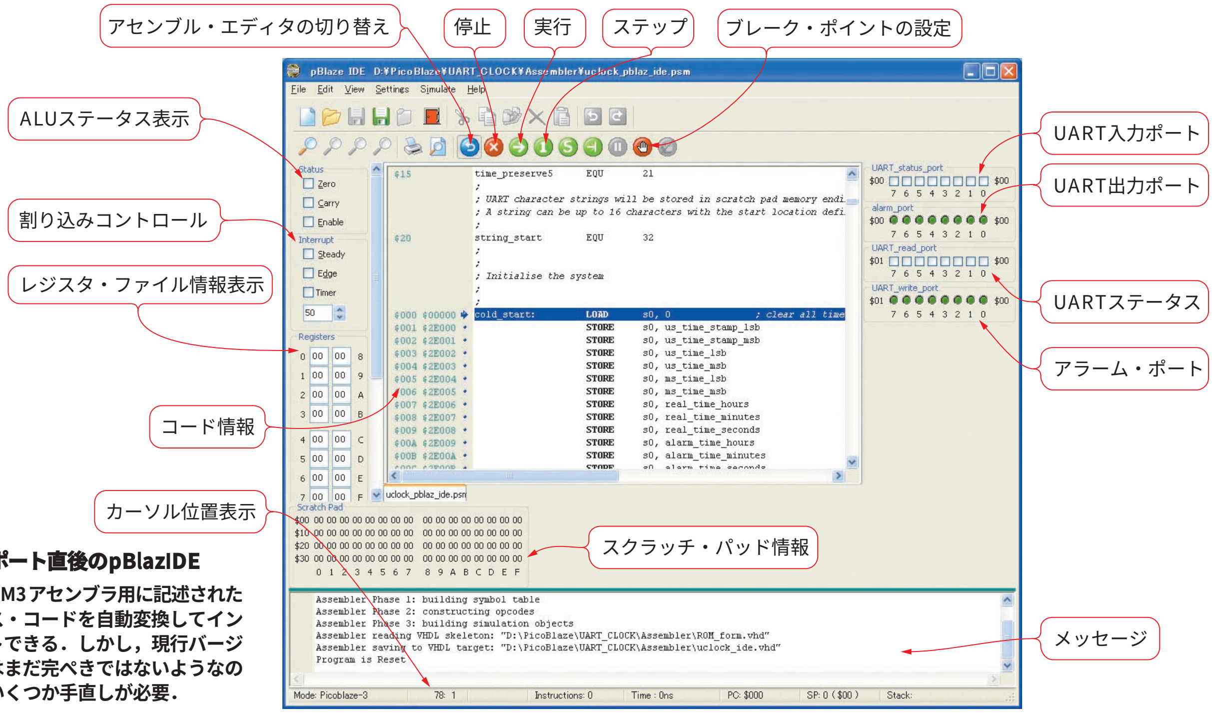


図8
インポート直後のpBlazIDE
KCPSM3アセンブラ用に記述されたソース・コードを自動変換してインポートできる．しかし，現行バージョンはまだ完ぺきではないようなので，いくつか手直しが必要．

コラム　pBlazIDEのためのソース・コードの変更

pBlazIDEでは，定数やアドレス定義の記述方法が標準のPicoBlazeと異なります．例えば，PicoBlazeでは定数の定義を，

```
CONSTANT label1, 01
```

と記述します．一方，pBlazIDEでは，

```
label1  EQU $01
```

と記述します．

pBlazIDEにはインポート時の自動変換機能があるので，基本的には何もしなくてもかまわないはずです．しかし実際には，まだ自動変換機能が完ぺきに働かないこともあるようです（図A-1）．今回のサンプル・コードでもエラーとなりました．

エラー・メッセージが出ないようにするには，いくつかプログラムを変更する必要があります．以下に，原因とサンプル・コードの修正方法を説明します．

1）pBlazIDEは大文字と小文字を認識しない

uclock.psmでは定数でcharacter_aとcharacter_Aが使用されていますが，pBlazIDEではこれらを同一の変数とみなすため，エラーになります．したがって，character_Aのほうをcharacter_l_aのように変更する必要があります．エディタの置換コマンドを使用し，AからZまでを修正します．

2）先頭が数字のラベルはエラー

uclock.psmでは，1char_to_valueと2char_to_valueの2種類のラベルが使用されています．これは，pBlazIDEではエラーとなります．これらのラベルをそれぞれ，one_char_to_value，two_char_to_valueのように先頭に数字を使用しないラベルに変更します．

3）FETCH，STOREコマンドの記述が少し違う

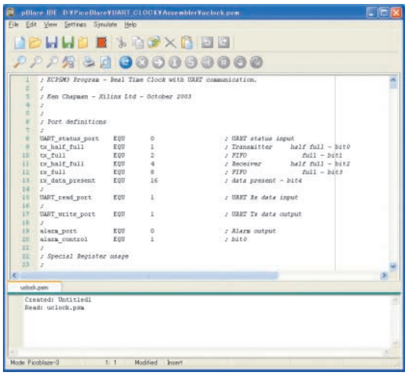
KCPSM3アセンブラでは，スクラッチ・パッドへのアクセスにおいてレジスタ・ファイルを使用する場合，以下のように記述します．

```
FETCH s0, (s1);
STORE s0, (s1);
```

pBlazIDEはかっこ（）が必要ないので，

```
FETCH s0, s1
STORE s0, s1
```

のように記述を変更します．つまり，かっこを削除するのです．



図A-1
自動変換の画面

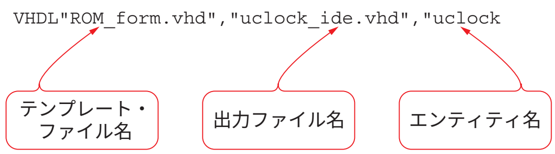


図9　pBlazIDEでPicoBlaze用のROMファイルを作成

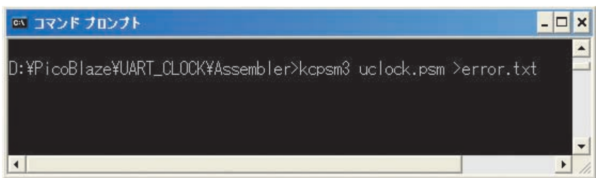


図10 PicoBlazeアセンブラ(KCPSM3)の実行
コマンドプロンプトで実行する.

リスト4　I/Oポートの定義

```
UART_status_port    DSIN    0    ; UART status input
UART_read_port      DSIN    1    ; UART Rx data input
UART_write_port     DSOUT   1    ; UART Tx data output
Alarm_port          DSOUT   0    ; Alarm output
```

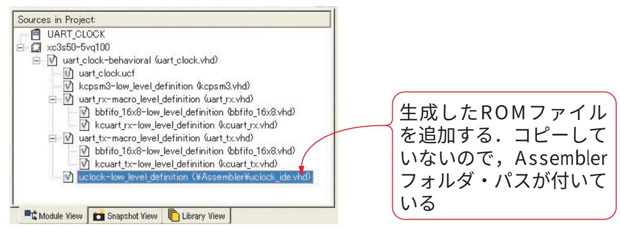


図11　ソフトウェアのプロジェクトへの登録
ツール・バーから「Project」→「Add source」でファイルを選択する.

ソース・コードにいくつかの情報を追加することと，I/Oポート記述の修正を行う必要があります．

pBlazIDEでPicoBlaze用のROMファイルを作成するときには，テンプレート・ファイル名と出力ファイル名，エンティティ名を入力します(**図9**)．

pBlazIDE上で入出力ポートのシミュレーションを行うには，**リスト4**のように入出力ポートであることを明示しなければなりません．入力はEQUの代わりにDSIN，出力はDSOUTを使用します．これらを変更しても，最終的に生成されるコードはEQUを使用した場合と同じなので，問題はありません．わずらわしいようですが，エディタの機能を使えば10分程度で直せる範囲の修正です．シミュレーション機能を使うことを考えると，めんどうでもこちらを使用することをお勧めします．

シミュレーションの実行にあたっては，まず，ブレーク・ポイントを設定します．ツール・バーから「Simulate」→「Toggle」でセットします．ここでは，933行めにセットします(カーソルの位置はウィンドウの左下に出ている)．

ツール・バーから「Simulate」→「Run」をクリックしてプログラムを実行します．

割り込みを発生させたいときは，左側のInterruptにあるEdgeにチェックを入れます．すると，ブレーク・ポイントを設置した場所は割り込みルーチンなので，その場所でプログラムが停止していることを確認できます．あとは，ステップ実行などを行いながら，割り込み時のふるまいを確認します．

このように，手動で割り込みを発生させたり，入力ポートのデータを操作したりすることでプログラムのふるまいを実機に入れる前に確認することができるので事前検証に効果的です．

4) アセンブル

シミュレーションによって動作を確認できたら，ソフトウェア・ソースをPicoBlazeで実行可能なオブジェクト・コードに変換するため，アセンブラ(KCPSM3)を実行します．

Windowsの「コマンドプロンプト」を起動し，**図10**のように実行します．ISE WebPACKに登録するUCLOCK.vhd(またはUCLOCK.v)が自動的に作成されます．pBlazIDEのアセンブル実行でもファイル作成が可能です．**図9**で指定したファイルが生成されているはずです．

5) ソフトウェアのプロジェクトへの登録

でき上がったROMファイルをプロジェクトへ追加します．ソフトウェアは修正されることが多いので，プロジェクト・フォルダにコピーせず，生成したROMファイルのあるフォルダから直接登録したほうがよいでしょう．

ISE WebPACKのツール・バーから「Project」→「Add source」を実行してファイルを選択します(**図11**)．

6) ピン配置の定義 ── UCFファイルの登録

FPGAのピン配置の定義には，拡張子が.UCFのファイルを使います．ツール・バーの「Project」→「Add New source」→「Implementation Constraints File」を選択します(**図12**)．作成するファイルがどのモジュールに属するかを決めるので，最上位ファイルのエンティティ名を選択します．これで空のUCFファイルが作成されます．エディタを開いて，**図12(d)**のように編集して完了です．

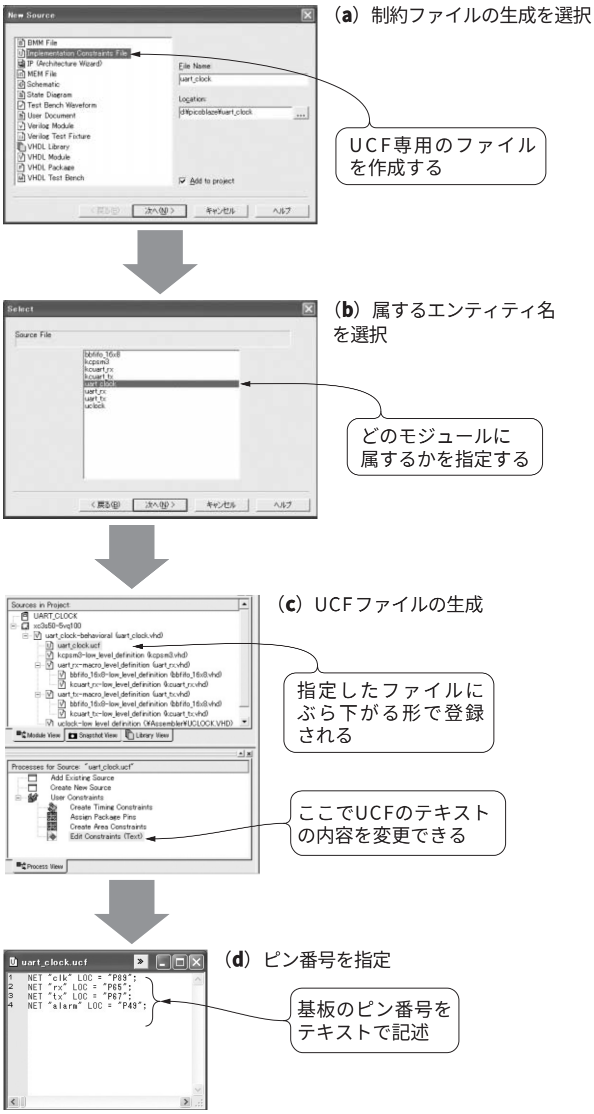


図12　ピン配置の定義
UCFファイルに記述する．

7）ビットストリームの作成

論理合成と配置配線を行い，FPGAに書き込むビットストリーム・データを作成します．

ISE WebPACKの左側にあるプロセス・ウィンドウの「Generate Programming File」をダブルクリックするだけです．ただし，初期設定のままではコンフィグレーションROM用のビットストリームになってしまいます．付属FPGA基板には，標準ではコンフィグレーションROMが搭載されていないので，JTAGでダウンロードして動作するビットストリームを生成します．「Generate Programming File」上でマウスを右クリックして「プロパティ」を選択し，Startup Optionタブの「FPGA Start-Up Clock」を「JTAG Clock」に変更します（**図13**）．

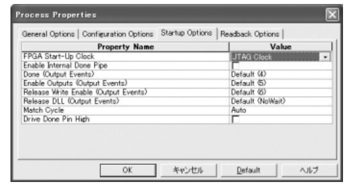


図13　ビットストリーム作成時の設定
TAGでダウンロードして動作するビットストリームを生成する場合の設定．



図14　ハイパーターミナルのASCII設定ダイアログ

表2　UART時計のコマンド

動　作	ターミナル・コマンド
時間表示	Time
時間設定（時：分：秒）	Time hh:mm:ss
アラーム設定（時：分：秒）	Alarm hh:mm:ss
アラームON	Alarm on
アラームOFF	Alarm off

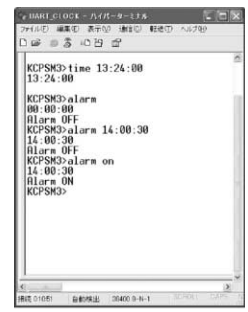


図15　ハイパーターミナルで時計を動作させたようす

8）実機による動作確認

ビットストリーム・データが生成されたら，JTAG経由でダウンロードします．

次に，パソコンでシリアル通信機能を使用できるターミナル・ソフトウェア（例えば，Windowsのハイパーターミナル）を起動します．設定は「8ビット・データ長，38,400bps，パリティなし，ストップ・ビット1ビット，フロー制御なし」とします．ハイパーターミナルを使用する際，ASCII受信設定オプションで「着信データに改行コードを付ける」をチェックする必要があります（**図14**）．

ハイパーターミナルから**表2**に示すコマンドを入力して，時計の動作が確認できれば動作確認終了です（**図15**）．

たなか・りょうへい
(株)ダイヘン
tanaka@advanced-ip.jp

<筆者プロフィール>
1971年生まれの33歳．同志社大学 工学部卒．現在はFPGAにソフトCPUコアを搭載し，ディジタル信号処理関連の製品開発を行っている．今回の記事執筆を通じてPicoBlazeの開発者であるKen Chapman氏と知遇を得てうれしく思っている．

5